Package 'ggpointless'

October 13, 2022

Title A Small Collection of Geometries, and Stats for 'ggplot2'

Version 0.0.3

Description A collection of geometries and stats for 'ggplot2'. Currently it supports geom_pointless() which adds minimal emphasis to your plots. Or just some visual sugar. geom_lexis() draws a 45° lifeline of an event that mimics lexis diagrams. geom_chaikin() applies Chaikin's corner cutting algorithm to a path.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.1

URL https://flrd.github.io/ggpointless/, https://github.com/flrd/ggpointless

BugReports https://github.com/flrd/ggpointless/issues

Depends ggplot2 (>= 3.3.0), R (>= 3.3)

Suggests knitr, covr, testthat (>= 3.0.0), rmarkdown, ragg, scales, ggtext, ggrepel, vdiffr (>= 1.0.0)

Config/testthat/edition 3

LazyData true

VignetteBuilder knitr

Collate 'aaa.R' 'data.R' 'geom-chaikin.R' 'legend-draw.R' 'geom-lexis.R' 'geom-pointless.R' 'ggpointless-package.R' 'stat-lexis.R' 'stat-pointless.R' 'utils.R'

NeedsCompilation no

Author Markus Döring [aut, cre]

Maintainer Markus Döring <m4rkus.doering@gmail.com>

Repository CRAN

Date/Publication 2022-08-25 08:22:40 UTC

R topics documented:

`co2_ml` *Monthly CO2 records taken at Mauna Loa, since March 1958*

Description

Atmospheric Carbon Dioxide Dry Air Mole Fractions from the NOAA GML Carbon Cycle Cooperative Global Air Sampling Network. Monthly time series constructed from daily mean values, from March 1958 to January 2022.

Usage

```
co2_ml
```

Format

A data frame with 766 rows and 5 variables

date date of measurement

year year of measurement

month month of measurement

co2_ppm CO2 concentration, in parts per million

decade decade of the measurement

Source

Dr. Pieter Tans, NOAA/GML (gml.noaa.gov/ccgg/trends/) and Dr. Ralph Keeling, Scripps Institution of Oceanography (scrippsco2.ucsd.edu/).

https://gml.noaa.gov/ccgg/trends/data.html

covid_vac *Rates of COVID-19 Cases and Deaths by Vaccination Status*

Description

Data on overall weekly rates of COVID-19 cases and deaths among fully vaccinated and unvaccinated people aged 12 years and older, according to COVID-19 positive specimen collection date. Data covers the periods from April 4, to December 25, 2021.

Usage

```
covid_vac
```

Format

A data frame with 146 rows and 4 variables

date Week of data collection

incidence COVID-19 cases and deaths, standardized by age

status vaccination status

outcome COVID-19 cases and deaths

Source

Centers for Disease Control and Prevention, Rates of COVID-19 Cases and Deaths by Vaccination Status

https://covid.cdc.gov/covid-data-tracker/#rates-by-vaccine-status

female_leaders *Female leaders of independent states.*

Description

Data from Wikipedia on women who have been elected or appointed head of state or government of their respective countries since the interwar period (1918–1939).

Usage

```
female_leaders
```

Format

A data frame with 131 rows and 5 variables

name Person

startdate Start of tenure

enddate End of tenure

country Country

power Executive or non-executive

Details

This list includes women who were appointed by a governing committee or parliament where heads of state or government are not directly elected by citizens. The list does not include women chosen by a hereditary monarch.

Source

wikipedia.org

https://en.wikipedia.org/w/index.php?title=List_of_elected_and_appointed_female_heads_of_state_and_government&oldid=1078024588

---

geom_chaikin *Apply Chaikin's corner cutting algorithm to smooth a path*

Description

Chaikin's corner-cutting algorithm can be used to smooth sharp corners of a path.

Usage

```
geom_chaikin(
  mapping = NULL,
  data = NULL,
  stat = "chaikin",
  position = "identity",
  ...,
  iterations = 5,
  ratio = 0.25,
  closed = FALSE,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_chaikin(
  mapping = NULL,
```

```
  data = NULL,
  geom = "path",
  position = "identity",
  ...,
  iterations = 5,
  ratio = 0.25,
  closed = FALSE,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes() or aes_(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
...	Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.
iterations	Integer. Number of iterations to apply. Must be between 0 and 10.
ratio	Numeric. Cutting ratio must be between 0 and 1.
closed	Logical. Specify if result is an open or closed shape.
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders().
geom, stat	Use to override the default connection between geom_chaikin and stat_chaikin.

Details

Chaikin's corner cutting algorithm iteratively turns a jagged path into a smooth path.

The recursion formula starts from two vertices A and B, which represent a single corner of your path. From this, the algorithm derives two new points: one at the specified ratio when going from point A to point B, and one when going from B to A in the opposite direction. By default, a ratio of 0.25 results in two points: the first at 25% of point A and the other at 75% of point A (or 25% of point B). Those new points form a smoother path. Then the algorithm applies the same rule to each pair of new points. The rule is applied iterations times. The maximum number of iterations is 10, default is 5.

The ratio parameter must be a number between 0 and 1. If ratio > 0.5, then it will be flipped to 1 - ratio, and a message is shown.

Aesthetics

geom_chaikin() understands the following aesthetics (required aesthetics are in bold):

- **x**
- **y**
- alpha
- color
- group
- linetype
- size

References

Chaikin, G. An algorithm for high speed curve generation. Computer Graphics and Image Processing 3 (1974), 346–349

Examples

```
set.seed(42)
dat <- data.frame(
  x = seq.int(10),
  y = sample(15:30, 10)
)

p1 <- ggplot(dat, aes(x, y)) +
  geom_line(linetype = "12")

p1 +
  geom_chaikin()

p1 +
  geom_chaikin(iterations = 1)

triangle <- data.frame(x = c(0, 0, 1), y = c(0, 1, 1))
p2 <- ggplot(triangle, aes(x, y)) +
```

```
  geom_path(linetype = "12") +
  coord_equal()

# ratio let's you control
p2 + geom_chaikin(ratio = .1)
p2 + geom_chaikin(ratio = .5)

# closed parameter to generate a closed shape - or not
p2 + geom_chaikin(iterations = 5, ratio = 0.25, closed = FALSE) # default
p2 + geom_chaikin(closed = TRUE)
```

---

geom_lexis *Display events of different cohorts in form of a lexis charts*

Description

This geom can be used to plot 45° lifelines for a cohort. Lexis diagrams are named after Wilhelm Lexis and used by demographers for more than a century.

Usage

```
geom_lexis(
  mapping = NULL,
  data = NULL,
  ...,
  point_show = TRUE,
  point_colour = NULL,
  point_size = NULL,
  gap_filler = TRUE,
  lineend = "round",
  linejoin = "round",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_lexis(
  mapping = NULL,
  data = NULL,
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes() or aes_(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
...	Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.
point_show	logical. Should a point be shown at the end of each segment? TRUE by default
point_colour	color of a point
point_size	the size of a point
gap_filler	logical. Should gaps be filled? TRUE by default
lineend	line end style (round, butt, square)
linejoin	line join style (round, mitre, bevel)
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders().

Details

This geom draws 45° lines from the start to the end of a 'lifetime'. It is a combination of a segment, and a point. Besides y and yend coordinates this geom creates one additional variable called type in the layer data. You might want to map to an aesthetic with ggplot2::after_stat(), see Examples section and vignette("ggpointless") for more details.

Rows in your data with either missing x or xend values will be removed (your segments must start and end somewhere).

Aesthetics

geom_lexis() understands the following aesthetics (required aesthetics are in bold):

- **x**

- **xend**
- alpha
- color
- fill
- group
- shape
- size
- linetype
- stroke

Examples

```
df1 <- data.frame(
  key = c("A", "B", "B", "C", "D", "E"),
  start = c(0, 1, 6, 5, 6, 9),
  end = c(5, 4, 10, 9, 8, 11)
)
p <- ggplot(df1, aes(x = start, xend = end, color = key))
p +
  geom_lexis()
p +
  geom_lexis(gap_filler = FALSE)
p +
  geom_lexis(aes(linetype = after_stat(type)),
    point_show = FALSE
  )

# change point appearance
p + geom_lexis(
  point_colour = "black",
  point_size = 3,
  shape = 21,
  fill = "white",
  stroke = 1
)

# missing values will be removed
df2 <- data.frame(
  key = c("A", "B", "B", "C", "D"),
  start = c(0, 1, 7, 5, 6),
  end = c(5, 4, 13, 9, NA)
)
ggplot(df2, aes(x = start, xend = end, color = key)) +
  geom_lexis()

# Ideally, `x` values should be increasing, unlike
# in the next example
df3 <- data.frame(x = Sys.Date() - 0:2, xend = Sys.Date() + 1:3)
ggplot(df3, aes(x = x, xend = xend)) +
```

```
  geom_lexis()

# If `x` is of class Date, `xend` can't be of class `POSIXt` or
# `POSIXct`. The error is thrown by the `scales::date_trans` function.
## Not run:
ggplot(
  data.frame(x = Sys.Date(), xend = Sys.time()),
  aes(x = x, xend = xend)
) +
  geom_lexis()

## End(Not run)
```

`geom_pointless` *Emphasize some observations with points*

Description

This is a wrapper around `geom_point()` with the one additional argument: `location`. It allows to emphasize some observations, namely the first, the last, the minima and/or maxima, see examples. This geom is not particularly useful on its own, hence its name, but hopefully in conjunction with `geom_line()` and friends.

Usage

```
geom_pointless(
  mapping = NULL,
  data = NULL,
  stat = "pointless",
  position = "identity",
  ...,
  location = "last",
  na.rm = FALSE,
  orientation = NA,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_pointless(
  mapping = NULL,
  data = NULL,
  geom = "point",
  position = "identity",
  ...,
  location = "last",
  na.rm = FALSE,
  orientation = NA,
```

```
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes() or aes_(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
...	Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.
location	A character vector specifying which observations to highlight, default is "last".
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
orientation	The orientation of the layer. The default (NA) automatically determines the orientation from the aesthetic mapping. In the rare event that this fails it can be given explicitly by setting orientation to either "x" or "y". See the Orientation section for more detail.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders().
geom, stat	Overwrite the default connection between geom_pointless() and stat_pointless().

Details

The location argument allows you to specify which observations should be highlighted. If location is "last", the default, a single point will be plotted at the last non-missing observation. The locations are determined in the order in which they appear in the data – like geom_path() does compared to geom_line(). See the vignette("ggpointless") for more details.

Overplotting

Points may be plotted on top of one another. If `location` is set to `"all"`, then the order in which points are plotted from top to bottom is: `"first"` > `"last"` > `"minimum"` > `"maximum"`. Otherwise, the order is determined as specified in the `location` argument, which also then applies to the order legend key labels, see examples.

Orientation

This geom treats each axis differently and, can thus have two orientations. Often the orientation is easy to deduce from a combination of the given mappings and the types of positional scales in use. Thus, ggplot2 will by default try to guess which orientation the layer should have. Under rare circumstances, the orientation is ambiguous and guessing may fail. In that case the orientation can be specified directly using the orientation parameter, which can be either "x" or "y". The value gives the axis that the geom should run along, "x" being the default orientation you would expect for the geom.

Aesthetics

geom_pointless() understands the following aesthetics (required aesthetics are in bold):

- **x**
- **y**
- alpha
- color
- fill
- group
- shape
- size
- stroke

Computed variables

location locations, returned as factor

Examples

```
x <- seq(-pi, pi, length.out = 100)
y <- outer(x, 1:5, FUN = function(x, y) sin(x * y))

df1 <- data.frame(
  var1 = x,
  var2 = rowSums(y)
)

# not terribly useful on its own ...
p <- ggplot(df1, aes(x = var1, y = var2))
p + geom_pointless()
p + geom_pointless(location = "all")
```

```
# ... but in conjunction with geom_line(), hopefully
p <- p + geom_line()
p + geom_pointless()
p + geom_pointless(location = c("first", "last"))
p + geom_pointless(location = c("minimum", "maximum"))
p + geom_pointless(location = c("all"))

# The layer computes one additional variable, 'location',
# that you can map e.g. to the color aesthetic
p + geom_pointless(
  aes(color = after_stat(location)),
  location = c("all"),
  size = 3
)

# Example with missing first and last observations
set.seed(42)
df1 <- data.frame(x = 1:10, y = c(NA, sample(1:8), NA))
ggplot(df1, aes(x, y)) +
  geom_line() +
  geom_pointless(location = c("first", "last"))

# Change the order in which points are drawn when they overlap
df1 <- data.frame(var1 = 1:2, var2 = 1:2)
cols <- c(
  "first" = "#f8766d",
  "last" = "#7cae00",
  "minimum" = "#00bfc4",
  "maximum" = "#c77cff"
)

p <- ggplot(df1, aes(x = var1, y = var2)) +
  geom_path() +
  coord_equal() +
  # makes comparision easier
  scale_color_manual(values = cols)

# same as location = 'all'
p + geom_pointless(aes(color = after_stat(location)),
  location = c("first", "last", "minimum", "maximum")
) +
  labs(subtitle = "same as location = 'all'")

# reversed custom order
p + geom_pointless(aes(color = after_stat(location)),
  location = c("maximum", "minimum", "last", "first")
) +
  labs(subtitle = "custom order")

# same as location = 'all' again
p + geom_pointless(aes(color = after_stat(location)),
  location = c("maximum", "minimum", "last", "first", "all")
```

```
) +
  labs(subtitle = "same as location = 'all' again")

# Use stat_pointless() with a geom other than "point"
set.seed(42)
df1 <- data.frame(x = 1:10, y = sample(1:10))
ggplot(df1, aes(x, y)) +
  geom_line() +
  stat_pointless(
    aes(yintercept = y, color = after_stat(location)),
    location = c("maximum", "minimum"),
    geom = "hline"
  )

# Example using facets
# https://stackoverflow.com/q/29375169
p <- ggplot(economics_long, aes(x = date, y = value)) +
  geom_line() +
  facet_wrap(~variable, ncol = 1, scales = "free_y")

p +
  geom_pointless(
    aes(color = after_stat(location)),
    location = c("minimum", "maximum"),
    size = 2
  )
```

Index